**Project 8 Part 2 Rotating a platonic solid with OpenCV (perspective)**

Name: Arnav Jain Period: 3 Date: 05/04/2022

Did you name your file l082.cpp (Lower case L, then 082)? yes

Does your file compile & run on terminals? yes

Did you use a rotation matrix? yes

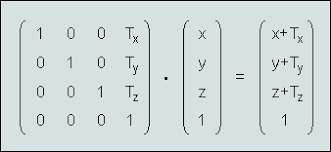
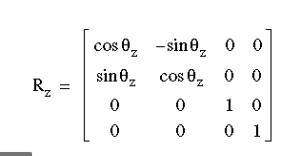
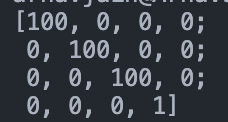
Did you start from the coordinates I provided for the cube?? yes

Describe here in words all the transformations you applied to vertices, for each describe how you implemented it in your code (by multiplying with a matrix, what was the matrix, or by adding a matrix, what was that matrix… be specific):

I scaled up by multiplying the coordinates by 100 using an homogenous matrix (#1),

After that I multiplied by the coordinates by a rotation about the z axis (#2), incrememting the angle by a degree for each of the 180 frames.

Lastly, I multiplied by the homogenous translation matrix (#3), which shifted it 400 to the right, and 300 down, placing the origin of the coordinate plane at the center of the video.



Describe in words the rotation you did: I rotated around the z-axis 180 degrees

Did you use homogenous coordinates? yes

(that allows you to combine all transformations into one matrix)

Did you combine all those transformations into one single matrix? no

If you used only one transformation matrix, what was it?

Did you do a perspective rendering? yes

What is the position of the eye you used? 1200

What is the plane of the screen you projected on? 500

Did you name your video rotation.avi? yes

What functions/methods from OpenCV did you use?

I used Mat for the matrices, VideoCapture to convert the matrices to a .avi video, a line and circle to represent vertices and edges, Points to represent the vertices, and Scalars/Vec3b to represent colors.

What functions/methods from OpenCV did you experiment with but ended not using?

I experimented with the \* operator to multiply matrices, but I decided to implement my own matrix multiplication method to get more experience with accessing and manipulating values in the Mat matrix type.

Obs.: feel free to rotate any platonic solid, around any line, and you may put the position of the screen/viewing window in any place as long as the rotating platonic solid can be seen reasonably.